Task-Oriented Object Tracking in Large Distributed Camera Networks

Eduardo Monari and Kristian Kroschel
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation
Fraunhoferstr. 1, D-76131 Karlsruhe, Germany
eduardo.monari@iosb.fraunhofer.de

Abstract

In this paper a task-oriented approach for object tracking in large distributed camera networks is presented. This work includes three main contributions. First a generic process framework is presented, which has been designed for task-oriented video processing. Second, system components of the task-oriented framework needed for the task of mult-camera person tracking are introduced in detail. Third, for an efficient task-oriented processing in large camera networks the capability of dynamic sensor scheduling by the multi-camera tracking processes is indispensable. For this purpose an efficient sensor selection approach is proposed.

1. Introduction

Several systems for automated multi-camera video surveillance have been developed by research groups during the last years (e.g. [1, 2, 3, 4], just to name a few), to enable object monitoring in large distributed camera networks. However, these systems aim to extract and analyze all extractable information from all available sensors. In many real-life applications however, only a subset of objects in a monitored area has to be monitored (e.g. external workforce or guests inside an industrial site) and subsequently only a small subset of data has to be processed to fulfill the surveillance task. For such applications classical approaches lead to unjustified waste of computational resources. In this paper we will introduce a new approach, called task-oriented data processing, by an exemplary video surveillance task - the multi-camera object tracking. To fulfill such a task in a task-oriented way, the challenge is to track preselected persons over several cameras, using task-relevant sensors and subsequently process task-relevant data only.

In Section 2 we will start with the introduction of our task-oriented process framework. It consists of distributed processes, which are described in Section 3 and 4 in detail. In Section 5 tracking results as proof of concept are shown and the advantage of the task-oriented concept in respect of the network and computational load is presented.

2. The Task-oriented Process Architecture

The tracking system described in this paper consists of a process framework designed for tracking of a single objects (e.g. persons) over different cameras. However, because there can be more than one single-person tracking-tasks running at the same time, the tracking system has to be able to handle an arbitrary number of tracking processes, simultaneously. Thus, from the internal point of view of the system, there is a need for a suitable distributed process architecture, which allows for dynamic instantiation of tracking sub-tasks and task-sensor-assignment.

As a solution for the internal structure of such a system, in [5] an architecture, for task-oriented data processing, has been proposed (sketched in Fig. 1). In this architecture, task-oriented in particular means, that the system focuses on task related events or objects only, and subsequently does not observe all objects in view and does not process all sensors simultaneously. The idea hereby was to design a framework, in which autonomous processes are charged with the specific surveillance tasks (here, tracking of a single object). Tracking agents then try to fulfill the dedicated task by self-organization. In particular, the agents autonomously determine and process task-relevant sensors, to minimize network and computational load.

The process architecture consists of a logical structure with one fixed and one dynamic level. The lower level (fixed) summarizes the data and information sources (sensors) with associated Intelligent Vision Platforms (IVPs).
Figure 2. Information flow between the sensor oriented IVPs, and a single task-oriented PRC for single target tracking.

Each sensor is processed by a dedicated IVP, which has the capability of self-localization, motion detection and feature extraction. That is, each IVP is able to determine the position and appearance information of all objects in view and is additionally able to provide information about the camera field-of-view.

The dynamic level consists of temporarily existing tracking agents, the so-called Processing Clusters, or PRCs in short. The PRCs are advanced processes, which are identified with a specific surveillance task (here, person tracking). A PRC includes two sub-modules - the so-called Tracking Module and the Dynamic Sensor Manager (Fig. 2). While the Tracking Module is responsible for data association and fusion of observations provided by task relevant sensors (so-called cluster sensors), the Dynamic Sensor Manager incorporates a knowledge-based camera selection algorithm for determination of the cluster sensors. Once, the task-relevant cameras are determined, the PRCs subscribe to them. In doing so, a PRC manages and process multiple sensors and additionally, is able to autonomously determine those IVPs, needed to fulfill the associated surveillance task (Fig. 2). Furthermore, performing the dynamic sensor selection, only task-related sensor data is processed by the PRC.

Now, in the following sections we will describe the introduced components (PRC and IVP) in detail, one by one. We start with the video analytics integrated as "Person Detectors" on the IVPs. Hereby, we describe the object detection functionalities used for person detection, and the extracted features for object description.

In the section after, the multi-camera object tracking performed by a Tracking-PRC is described.

3. Intelligent Vision Platform

The Intelligent Vision Platforms are distributed software applications with detection and feature extraction capabilities only. Each camera in the network is directly assigned to a dedicated IVP. In case of smart cameras, with embedded onboard processing capabilities, the IVP is the onboard processing framework (Fig. 2a). Depending on the available computational power of the intelligent sensors or the specific detection task ordered by a PRC, the IVP activates one or multiple video analytic plug-ins. All activated plug-ins perform individual algorithms for event detection and feature extraction (e.g. motion detection, face detection, change detection (abandoned luggage, theft)). After this the extracted information and object features are available to all subscribed PRCs. By providing observation descriptions on feature level, and additionally only on demand the network load is reduced significantly. This guarantees that only task-relevant processes affects network load by extensive data exchange.

The IVPs as intelligent sensors are performing video content description only. That is, no task-related decision making is performed by the distributed smart cameras. In doing so, the description of the observations on feature level can be used by different PRCs which are assigned with different surveillance tasks.

For the person tracking task, a person detection plug-in is applied to each IVP. Each of them performs a background estimation and subtraction algorithm for motion detection with additional post-processing for segmentation improvement (shadow detection, disturbance data removal). For background estimation we use a slightly modified \( \Sigma \Delta \) Approach as described in [6]. The variation of the gray value of each pixel is described as a single Gaussian, using a recursive estimation algorithm. The foreground objects are extracted using a background subtraction approach with a segmentation enhancement using a pyramidal spatial Markov model for fast dynamic noise reduction [6]. The object segments are represented by a blob image determined by a connected component analysis.

To each blob (object segment) geometrical attributes are extracted (object height, width, aspect ratio and compactness). If the geometric attributes of a blob matches human-like attributes, then additional evaluation algorithms are applied (e.g. shadow detection, feature extraction for object description).

3.1. Extraction of Spatial Features

Given a blob (object segment), detected by a camera, the spatial features are calculated using available camera calibration parameters. The object position \( p \) in world coordinates is estimated by ray intersection of pixel \( p(x, y) \) with the planar ground plane (as shown in Fig.3). Four more ray intersections \( r_1, r_2, r_3, r_4 \) are calculated for the pixels \( p_1(x, y - \delta_y), p_2(x, y + \delta_y), p_3(x - \delta_x, y), p_4(x + \delta_x, y) \). Due to the non-linear relationship between 2D image and 3D world coordinates, the position deviation along the camera orientation axis is approximated by \( \sigma_1 = |r_1 - r_2|/2 \) and by \( \sigma_2 = |r_3 - r_4|/2 \) for the devia-
tion horizontally to the camera orientation. Given the position $r_{cam}$ of the camera in world coordinates the orientation of the approximated bivariate Gaussian distribution is given by $\phi = \angle(r_{cam} - r)$.

Finally, we can summarize the spatial features for an object $j$, detected by camera $i$ as: $f^{(sp)}_{i,j} = [x_{i,j}, y_{i,j}, \sigma_{1_{i,j}}, \sigma_{2_{i,j}}, \phi_{i,j}]^T$ and re-describe the measurements as

$$z = (x, y) \quad \text{and} \quad \Sigma = Rot^T \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix} Rot$$

with $Rot$ as the 2D rotation matrix given $\phi$, $z$ and $\Sigma$ modeling a new measurement as bivariate Gaussian distribution on the 2D ground plane.

3.2. Extraction of Appearance Features

For each detected object, a color descriptor is calculated in four steps as shown in Fig. 4: First, the image region containing the detected object is cut out of the image and rescaled to a normalized height (e.g. 128 pixels), and is then post processed by a color normalization (gray world assumption and histogram specification), whose parameters were priorly estimated over the whole image.

In a second step, the color space of the detected object is reduced using a mean-shift-based approach as described in [7]. Hereby, the color texture of the observed object is reduced to a small number of homogeneously colored body segments. In a third step, for each color segment, the area (in pixels) and the centroid are calculated and segments smaller than 5% of the body area are removed.

Finally, for approximated spatial description of the detected person, the body is subdivided in three sections: head, upper body and lower body. We define, starting from the bottom, the first 55% as lower body, the next 30% as upper body, and the remaining 15% as head. These assumptions are largely along the lines to known anthropometry standards. The appearance descriptor is now composed, by assigning the color segments to the corresponding body part by its centroids. Identical colors, which belongs to the same body part, are merged to one. In doing so, spatial relationships within a body part is lost but at the same time this approach leads to an invariant representation of the object, concerning to view angles of multiple cameras.

Let $F^{(app)} = (f^{(app)}_1, f^{(app)}_2, \ldots, f^{(app)}_Q)$ be the collection of all $Q$ color segments, with $f^{(app)} = [l, u, v, w, b]_T$, with $L, u, v$ as the chromaticity and luminance values of the homogeneous segment in CIE $L^*u^*v^*$ color space, $w \in \{0..1\}$ (weight) as the area fraction of the color segment relative to the object area and $bp = \{\text{head, upper body, lower body}\}$ as the body part index, the centroid of the segment belongs to.

Finally, the person detection can be summarized as follows: A person detection plug-in is continuously performing object detection and feature extraction for all observable objects in the dedicated video data. For each detected person $j$, an object descriptor $O_{i,j} = (f^{(sp)}_{i,j}, F^{(app)}_{i,j})$, with $i$ as the camera index, is available to all subscribed PRCs for further task-oriented multi-camera data evaluation.

4. The Processing Cluster (Tracking Agent)

PRCs are responsible for the task-oriented multi-sensor data processing and fusion (e.g. tracking) of a dedicated object in the scene. They include a so-called Dynamic Sensor Manager and the Tracking Module as their main components (Fig.2). While the Dynamic Sensor Manager is responsible for sensor scheduling and data request from task-relevant cameras, the Tracking Module is performing data evaluation and association to the data of those sensors. In following subsection we will describe the data association and fusion process performed by the Tracking Module, first. After that, the sensor selection approach integrated in the DSM will be introduced.

4.1. Tracking Module

The Tracking Module in principle consists of a slightly modified Kalman Filter for fusion of spatial features (position tracking) with an additional appearance based data association filter. The modification regards to an integrated exception handling, for using a linear Kalman Filter in a non-linear environment (like a building). Such exceptions occur, if due to linear combination of the predicted state es-
timate $\hat{x}$ and the measurement $\hat{z}$ leads to an invalid value of the posterior state estimate mathit{mathbf{x}} (e.g. a position inside a wall of the building). In such cases we re-initialize the Kalman Filter by the last measurement.

In a first step the spatial consistency of an object observation (given by its spatial feature set) is evaluated, and if the object classified as spatial consistent, an appearance based similarity metric is calculated for data association. Hereby, for similarity calculation of two appearance feature sets $F^{(app)}$ and $G^{(app)}$ the Earth Mover’s Distance (EMD) proposed by Rubner et al. in [8] is used. The EMD formulates the distance between two distributions as a transportation problem. Formally, our EMD for color similarity is calculated as

$$\text{EMD}(F^{(app)}, G^{(app)}) = \min \sum_{q} w_{qr} \Delta_{qr}$$

with $\Delta_{qr} \geq 0$ as the color distance and $w_{qr} \geq 0$ as the moved weight from color sample $q$ of $F^{(app)}$ to $r$ of $G^{(app)}$.

The EMD has shown good performance in image retrieval comparing color and texture histograms [8].

In our appearance similarity calculation, the feature space is defined by a modified three-dimensional L*u*v color space, expanded by an additional dimension for the body part. The L*u*v color space has been chosen for calculation of color differences $\Delta_{col}$ by the Euclidean distance. The weight of a feature is given by $\tilde{w}$, as described before.

After initialization and sensor selection, the Tracking Module receives feature vector sets $O_i$, $i \in S_k$ asynchronously, with $i$ describing the observing sensor in the actual camera cluster $S_k$. Due to the sequential processing discarding the camera index $i$ let $O_k = (O_{k,1}, O_{k,2}, \ldots, O_{k,m}) = (F_1^{(app)}, F_2^{(app)}, F_3^{(app)}, \ldots, F_m^{(app)})$ be a feature vector set, describing all $m$ detected objects at time $k$, by a single camera. The multi-camera single object tracking problem is now to determine if one (or which) of the observations provided by $O_k$ belongs to the selected object of interest $O_0$. Our data association algorithm is subdivided in five steps:

1. Given a prediction of the spatial object state $(\hat{x}, P^{-})$, with $\hat{x}$ as the predicted object position estimation and $P^{-}$ as the estimated error covariance, calculate the squared Mahalanobis Distance for $F_j^{(app)} \rightarrow z_{j,k}$:

$$d_j^{(mah)} = (z_{j,k} - \hat{x}_k)^T (P_k^{-})^{-1} (z_{j,k} - \hat{x}_k).$$

2. Determine a subset of observations, which are classified as spatial consistent (gating):

$$O'_k = \{ O_{j,k} \mid d_j^{(mah)} \leq \tau^{(mah)} \}.$$ 

3. For the subset $O'_k$ calculate appearance similarity coefficients by equation (2):

$$d_j^{(EMD)} = \text{EMD}(F_j^{(app)}, F_{0,t_0}^{(app)}).$$

4. Determine a subset of $O''_k$, which fulfill appearance similarity restrictions:

$$O''_k = \{ O_{j,k} \mid d_j^{(EMD)} > \tau^{(sim)} \}.$$ 

5. The feature set $O_{j,k}$ is now given by:

$$\hat{j} = \arg\max_{j \in O''_k} (d_j^{(EMD)}).$$

The proposed approach has been integrated in an experimental video surveillance system, installed in our lab. It consists of about 25 cameras installed on three floors in our building. All sensors are state-of-the-art IP-cameras, and commercially available. The fields of view of the sensors cover a major part of the test areas, but not completely.

The computational capacity of the experimental systems consists of 9 high-end PCs available on the market. Five of them emulate the intelligent cameras (with 5 running IVPs each). The other processor cores provide computational resources for PRCs which means for tracking processes.

Fig. 5 shows a generated trajectory as a result of the Tracking-PRC. Hereby, the PRC tracked a person over several cameras in our lab. The tracking approach shows good performance if the object to be tracked has significant color features, that is, the person can be identified by its color descriptor. In particular these features has to be unique for re-identification, if the person is not visible for a longer period of time due to non-overlapping sensor coverage. However, for our investigation on the task-oriented approach, these assumptions are sufficient, since the tracking approach is only for proof of concept.
4.2. Dynamic Sensor Management

The Dynamic Sensor Manager (DSM) module is an independent component which periodically determines sensors (cluster members) for a certain surveillance task. Given the actual object state estimate (that is the estimated actual object position) and the prior knowledge about sensors and the environment, the camera selection algorithm integrated in the DSM calculates the relevant subset of the sensor network that is needed for further observation of the object. In addition to determine the sensors with the estimated position in FoV as a first objective, a special goal of this algorithm is to determine the minimum number of sensors needed to relocate an object, even if the object is temporarily out of sight (e.g., by non-overlapping sensor coverage).

Depending on the determined sensors in the cluster, the PRC subscribes to cluster IVPs, which now in turn start providing most recent spatial and appearance data about observed objects. Each time new observation data is available, the data validation and association process is performed by the Tracking Module for object position estimation.

For an efficient determination and selection of the task-relevant cameras, a knowledge-based approach is used, which evaluates the geometrical relations between camera sensing ranges and building map given the actual object position. Hereby, for highly efficient evaluation of these relationships, the available knowledge about camera setup and building map is modeled in a special way. Thus, before introducing the camera selection algorithm, the knowledge modeling of the camera network is described.

4.2.1 Modeling of Camera Sensing Ranges

As described before, each camera or sensor in the network is assigned to a IVP. Additionally, self-localization and self-calibration capabilities are integrated in the IVPs, which means that each camera is able to provide a polygonal field-of-view (FoV) in world coordinates.

For our camera selection approach the so-called visibility polygons are needed. While a FoV represents the maximum observable area defined by the optical attributes of a camera, a visibility polygon is the subset of the FoV, which is directly observable, regarding environmental restrictions (e.g., occlusions by static objects, walls etc.).

However, by means of the FoV parameters and the prior known building map, it is possible to calculate the visibility polygon. In our system this is done by the visibility algorithm proposed in [9]. As a result of this processing step, for each camera \( s_i \) in the network, a visibility polygon \( \mathcal{L}_i = \{l_1, l_2, \ldots, l_j\} \) is determined.

It is important to mention, that for non-static sensors, the sensing ranges have to be kept up-to-date during processing, to guarantee correct sensor selection.

4.2.2 Modeling of Geometrical Prior Knowledge

A central point of our approach is the modeling of both, the environment (building map) and the camera visibility polygons, in a common geometrical 2D representation. The building model (Fig. 6a) is given by a set of \( p \) line segments \( \mathcal{M} = \{m_1, m_2, \ldots, m_p\} \) and for each camera \( s_i \in \mathcal{S} \) the sensing range is given by a visibility polygon (Fig. 6b), also represented by a set of line segments \( \mathcal{L}_i = \{l_1, l_2, \ldots, l_j\} \). All line segments are now used to induce a so-called arrangement of line segments \( A(\mathcal{L} \cup \mathcal{M}) = (\mathcal{E}, \mathcal{V}, \mathcal{F}) \), with \( \mathcal{L} \cup \mathcal{L}_i \) [10]. That is, \( \mathcal{E} \) is a set of edges, \( \mathcal{V} \) is a set of vertices and \( \mathcal{F} \) is a set of resulting faces in the plane (Fig. 6c). Hereby, a vertex is the intersection or an end point given a line segment of \( \mathcal{E} \cup \mathcal{L}_i \), an edge is the maximum portions of a line without any vertex in between and a face is a subset of the plane that doesn’t contain a point on an edge or a vertex. Consequently, a face is an open polygonal region whose boundary is formed by edges and vertices.

For efficient handling of arrangements or planar graphs, the doubly-connected edge list structure (DCEL) turns out to be a standard in computational geometry [10]. For DCELs and line arrangements there are highly efficient algorithms for point, edge or face localization and manipulation ([10, 11]). DCEL was chosen, since efficient local manipulation and point localization capability in large arrangements are enhancing the effectiveness of our approach.

4.3. Camera Selection Algorithm / Clustering

The object tracking is based on a modified Kalman filter for position estimation. Thus, by the integrated prediction filter, Tracking Module is able to provide a prediction about the object state (position on the ground plane) \( \hat{\mathbf{x}}(k|k-1), \mathbf{P}(k|k-1) \). In addition, given prior knowledge about sensors and the environment, the camera selection algorithm calculates the relevant subset of the sensor network that is needed for further observation of the object, even if the object is temporarily out of sight.

Let \( \mathcal{S} \) be a set of sensors distributed in an area under surveillance (e.g., as in Fig. 7a). The relevant sensors for multi-camera object tracking are called cluster sensors.
$S_k^{\text{cluster}}$. The proposed approach differs between three subsets of sensors in $S_k^{\text{cluster}}$: One subset of the cluster (the so-called active sensors) $S_k^{\text{active}}$ includes all sensors with the object of interest in view (Fig. 7b, red FoVs). These sensors, of course, are responsible for direct object observation. The second subset (so-called passive sensors) are those sensors (with no object in view) needed to limit the local area in the surrounding of the object under observation (Fig. 7b, yellow FoVs). The passive sensors are determined for object recognition, in case of object disappearance. Because the sensor selection algorithm guarantees that the object is surrounded by passive or active sensors all time, the observed object has to reappear in one of them. Subsequently, the tracking process has to request observation data from these sensors only.

However, while for multi-camera tracking the object observations are requested from active cameras all the time, only a subset of the passive sensors is temporarily relevant for the tracking task. A motion model can be defined for the object of interest. For our tracking approach a motion model has been defined as state transition matrix of the modified Kalman Filter integrated in the Tracking Module. Due to this motion model, only the subset of the passive cameras which can be reached by the object by the time elapsed since the last detection are included into the multi-camera tracking. For object relocation, only the observations from the active and this subset of the passive cameras are needed. So, communication paths have to be set up to these sensors only. This subset is defined by $S_k^{\text{com}}$ (Fig. 7, red FoVs).

Now, after introduction of these basic terms, we start with our sensor selection method. For each $s_i \in S$ a set of line segments $L_{i,k} = \{l_1, l_2, \ldots, l_j\}$ (visibility polygon on the 2D ground plane at time $k$) is given. The collection of the visibility polygons is summarized by $L_k = \{L_{1,k}, L_{2,k}, \ldots, L_{n,k}\}$. Additionally, let a set of line segments $M = \{m_1, m_2, \ldots, m_p\}$ represent the building map.

Given this prior information, the sensor selection algorithm starts with an initialization step:

$$S_0^{\text{active}} = \emptyset, S_0^{\text{passive}} = S_0^{\text{com}} = \emptyset$$

$$A_0(L_0 \cup M) = (E_0, V_0, F_0)$$

At this time, there is no initial object position available (tracker not initialized). Thus, $S_0^{\text{active}}$ and $S_0^{\text{com}}$ are empty sets. $S_0^{\text{passive}}$ includes all sensors in the network, because without position information no spatial containment of the object is possible. Additionally, the complete arrangement of line segments $A_0(L_0 \cup M)$ is generated, including the line segments of the environment model and all line segments from sensor visibility polygons. It should be noticed, that the full generation of the arrangement of line segments is performed in prior. Later updates for non-static cameras are performed only for cluster sensors.

Once the tracker has been initialized (by definition of the object to observe) a prediction about the object state estimate is provided by the Tracking Module and camera selection can be performed. The tracking process provides predictions about the object state estimate periodically. For each prediction a new sensor selection loop is performed. Because for the selection approach it is essential to evaluate valid information about camera sensing ranges, in step 1 an update of the visibility polygons is performed first. Hereby, only the cluster sensors determined in the previous selection loop are updated, and therefore only a small subset of the arrangement has to be recalculated. After update of the arrangement the set of active sensors are calculated first (step 2). Given the predicted object position estimate $\hat{x}_k^\text{c}$ by the Tracking Module, for all visibility polygons $L_{i,k}$ of the cluster cameras $S_k^{\text{cluster}}$ a point-in-polygon-test is performed. Hereby we use an implementation of the well-known ray casting algorithm as described in [12]. The sensors with the object in sensing range are declared as active sensors (Fig. 7b/e).
It is important to notice, that while for the very first selection loop, all sensors have to be evaluated ($\mathcal{S}_0^{\text{(passive)}} = \mathcal{S}$), in subsequent iterations, the point in polygon test is only applied on a highly reduced number of camera polygons ($\mathcal{S}_{k-1}^{\text{(cluster)}}$) and is therefore very efficient.

Once the active sensors are known, the passive sensors can be determined in a subsequent step (3). First, a slightly modified arrangement of line segments $A_k((\mathcal{L}_k \setminus \mathcal{L}_k^{(\text{active})}) ∪ \mathcal{M})$ is created. This is, the arrangement includes line segments given by the environment model $\mathcal{M}$ and the visibility polygons of all sensors in the network, but without the active ones.

The subtraction of the line segments of the active sensors is a crucial part of the algorithm. By excluding the visibility polygons of the active sensors it is guaranteed, that all remaining line segments can be considered as physical (environment model, e.g., walls) or virtual (entry edges of surrounding cameras) borderlines (Fig. 7f).

Sensor Selection Algorithm

For each object state prediction $(\hat{x}_k^-, P_k^-)$, do

1. update arrangement of line segments:
   (a) Determine edges of passive sensors to remove: $\mathcal{L}_{\text{update}}^{(\text{remove})} = \{\mathcal{L}_i \in \mathcal{L}_{k-1} | s_i \in \mathcal{S}_{k-1}^{(\text{passive})}\}$
   (b) Perform an update of cluster sensors. In case of a follow-up prediction, only passive sensors are updated. $\mathcal{L}_{\text{update}}^{(\text{update})} = \{\mathcal{L}_i \in \mathcal{L}_{k-1} | s_i \in \mathcal{S}_{k}^{(\text{passive})}\}$
   with $\mathcal{S}_{\text{update}}^{(\text{cluster})} = \mathcal{S}_{k-1}^{(\text{cluster})}$ if first prediction, else $\mathcal{S}_{\text{update}}^{(\text{cluster})} = \mathcal{S}_{k-1}^{(\text{cluster})} \setminus \mathcal{S}_{k-1}^{(\text{active})}$.
   (c) The updated arrangement is given by $A_k' = A(\mathcal{L}_k' ∪ \mathcal{M}), \mathcal{L}_k' = (\mathcal{L}_{k-1} ∪ \mathcal{L}_{\text{remove}}^{(\text{update})}) ∪ \mathcal{L}_{\text{update}}^{(\text{update})}$

2. determine the active sensors $\mathcal{S}_{\text{active}}^{(active)}$: $\mathcal{S}_{\text{active}}^{(active)} = \{s_i \in \mathcal{S}_{k}^{(\text{cluster})} | \hat{x}_k^- \cap \text{Face}(\mathcal{L}_i,k) \neq \emptyset\}$

3. Determine the passive sensors:
   (a) Manipulate the arrangement of line segments: $\mathcal{L}_k = \mathcal{L}_k' \setminus \mathcal{L}_k^{(\text{active})}$ $A_k = A(\mathcal{L}_k ∪ \mathcal{M}) = (\mathcal{E}_k, \mathcal{V}_k, \mathcal{F}_k)$
   (b) Find the face in $A_k((\mathcal{L}_k ∪ \mathcal{M})$, which includes the predicted object position: $\mathcal{F}_{\text{predicted}} = \{\mathcal{F}_u \in \mathcal{F}_k | \hat{x}_k^- \cap \mathcal{F}_u \neq \emptyset\}$
   (c) Determine the sensors involved in the boundary edges of the face $\mathcal{F}_{\text{predicted}}$: $\mathcal{S}_{\text{active}}^{(\text{passive})} = \{s_i \in \mathcal{S} | \mathcal{L}_{i,k} ∩ \text{CCB}(\mathcal{F}_{\text{predicted}}) \neq \emptyset\}$

4. Determine the plausibility-of-presence $\psi$ of the passive sensors and subscribe to relevant cluster sensors: $\mathcal{S}_{\text{active}}^{(\text{cluster})} = \{s_i \in \mathcal{S}_{k}^{(\text{cluster})} | s_i \in \mathcal{S}_{k}^{(\text{active})} \lor (\psi_{i,k} > \tau(\text{PoP}))\}$

The arrangement can also be constructed in a very efficient way, by temporarily removing the edges which belong to the actual active sensors. In doing so, even in very large sensor networks, the computational power needed for recalculation of the arrangement is minimized.

After manipulation of the arrangement $A_k'$ to $A_k$, the face $\mathcal{F}_{\text{predicted}}$ (containing the actual object position) is calculated. In a further step, the connected component boundary function CCB determines the edges which are incident to the face $\mathcal{F}_{\text{predicted}}$ (Fig. 7f, blue area). The edges of the CCB can derive from visibility polygons of the sensors ($\mathcal{L}_{i,k}$) or from the environment model $\mathcal{M}$. Because we are only interested in sensor selection, the edges deriving from the building map $\mathcal{M}$ are remain unconsidered, and the sensors involved in the CCB are added to $\mathcal{S}_{k}^{(\text{passive})}$.

The active and the passive sensors are now defined as the actual camera cluster $\mathcal{S}_{k}^{(\text{cluster})} = \mathcal{S}_{k}^{(\text{active})} \cup \mathcal{S}_{k}^{(\text{passive})}$.

The sensor selection process is now repeated from Step 2 periodically. It is important to mention, that this algorithm is obviously able to determine locally task-relevant cameras, by recursive determination of cluster members and an efficient manipulation of the arrangement of lines.

Now that the cluster sensors are determined, in a further step the temporarily relevant cameras $\mathcal{S}_k^{\text{com}} \subseteq \mathcal{S}_k^{\text{cluster}}$ are determined for request of observation data (step 4). This is done by evaluation of the plausibility-of-presence of the object of interest for each passive camera.

The plausibility-of-presence for $s_i \in \mathcal{S}_k^{\text{passive}}$ is defined as

$$\text{PoP}_i(\hat{x}_k^-, P_k^-) = \left(1 + \text{SPath}(\hat{x}_k^-, \mathcal{L}_{i,k}) / \sigma_1^{(\text{max})}(P_k^-)\right)^{-1},$$

with $\text{SPath}$ as the length of the shortest path from the predicted object position $\hat{x}_k^-$ to the visibility polygon of $s_i$ and $\sigma_1^{(\text{max})}$ as the worst-case assumption about the uncertainty of the predicted object state estimate (given by the covariance matrix $P_k^-$. For calculation of the shortest path we use Dijkstra’s algorithm [13]. Dijkstra’s algorithm, as a single-source-query-approach is able to calculate the shortest paths to all passive sensors simultaneously, $\sigma_1^{(\text{max})}$ represents the worst-case assumption about the object movement (position uncertainty).

In case of disappearance the prediction filter leads to an increase of the position uncertainty about the object, as defined by the motion model. So finally, if the plausibility-of-presence exceeds a predefined threshold, the reappearance of the object in these cameras is regarded as plausible. In this case, this sensor is added to $\mathcal{S}_k^{\text{com}}$, that is observation data is requested.

For quantitative evaluation of the sensor selection approach and investigation of the performance gain in large sensor network simulations have been performed. Fig. 8 shows the simulation results for the task of tracking an ob-
In this paper, an approach for single target tracking in large sensor networks by dynamic sensor selection has been presented. The process architecture mainly consists of distributed autonomous processes, which are responsible for tracking a dedicated person by evaluation of observation data, provided by the task-relevant IVPs. We have shown, that the PRCs are able to track single objects in a task-oriented manner, by combining their data association and fusion capabilities (Tracking Module), with an integrated camera selection approach (Dynamic Sensor Management). In doing so, the PRCs subscribe to task-relevant sensors and process task-related data and information, only.

The main contributions of the paper is the proof of concept of a task-oriented video and information processing. In our framework we divide the (sensor-related) detection and observation task, from the principal surveillance task, that is, the person tracking task. Furthermore, by performing a processing task combining the multi sensor data fusion approach with a dynamic sensor selection capability, we showed that a centralized process can be used also in very large sensor network.

5. Conclusions

In this paper, an approach for single target tracking in large sensor networks by dynamic sensor selection has been presented. The process architecture mainly consists of distributed autonomous processes, which are responsible for tracking a dedicated person by evaluation of observation data, provided by the task-relevant IVPs. We have shown, that the PRCs are able to track single objects in a task-oriented manner, by combining their data association and fusion capabilities (Tracking Module), with an integrated camera selection approach (Dynamic Sensor Management). In doing so, the PRCs subscribe to task-relevant sensors and process task-related data and information, only.

The main contributions of the paper is the proof of concept of a task-oriented video and information processing. In our framework we divide the (sensor-related) detection and observation task, from the principal surveillance task, that is, the person tracking task. Furthermore, by performing a processing task combining the multi sensor data fusion approach with a dynamic sensor selection capability, we showed that a centralized process can be used also in very large sensor network.

References
